

Um mich mehr mit dem Seam-Framework auseinanderzusetzen, werde ich die Webapplikation, Arien-Delikatessen für einen Bekannten entwickeln.

Die Anforderungen an ein Projekt das geeignet ist, sich mit dem Seam-Framework näher auseinanderzusetzen sind schnell beschrieben.

#### **Anforderungen:**

- Webbasiert
- Eingabe der Stammdaten und Modifikation der Stammdaten
- Buchungen der Bestellungen
- Erstellung der Rechnung mit Gutschrift
- Kundenrating für Discount und Auslieferungsform Vorkasse, etc)
- Erstellung der ToDo-Liste für die Mitarbeiter (Erstellung und Lieferung der Bestellungen)
- Lagerverwaltung
- Mahnsystem
- Erstellung div. Reports

Handelt es sich beim Seam-Framework doch um ein OpenSource- Projekt das die bekannten JEE Technologien (Hibernate, Spring, JSF, AJAX usw.) zur Verfügung stellt.



Abbildung 1

Als Layer zwischen Applikation und den div. JEE-Framework, vereinfacht Seam den Einstieg, Design und Portierung enorm.

Seam koordiniert zur Laufzeit die Interaktion zwischen Applikation und den unterschiedlichen Frameworks (Hibernate, AJAX, usw).

Die Beschäftigung mit den Besonderheiten der div. Layer entfällt, es braucht nur noch ein Framework gelernt werden.

(Abbildung 1)

Dazu bringt Seam einige Neuigkeiten wie Conversation, Page Flow, Business process, rule-based security, JavaScript (AJAX), PDF rendering, E-Mail composition, Charting, file uploads und Groovy Integration mit.

Seam ist mit den gängigen Applikationsservern wie BEA, OC4J, Tomcat, ClassFish, Websphere und natürlich JBoss kompatibel.

Ein weiterer, wichtiger technologischer Aspekt in Seam, ist in Abbildung 2 und 3 zusammengefasst

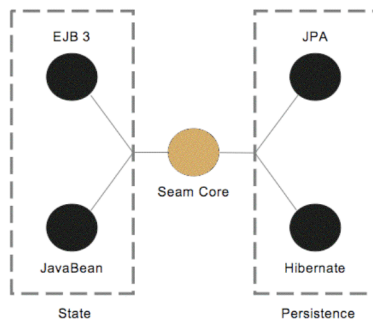


Abbildung 2

Der State stellt technology für behandeln der Applikationslogik und die Reaktion auf UI. Der persistente Schicht kümmert sich um Datentransport.

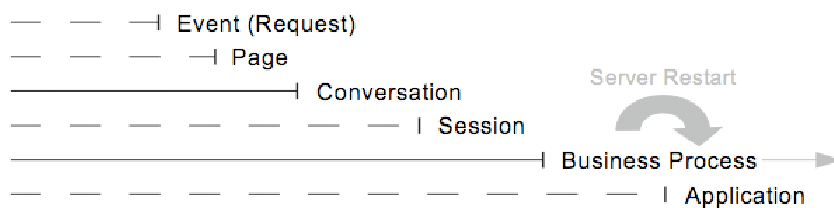


Abbildung 3

Für die weitere Informationen sieh die Links:

<http://www.seamframework.org/Documentation/KnowledgeBase>

<http://docs.jboss.org/seam/2.2.0.GA/reference/en-US/html/annotations.html>

<http://docs.jboss.org/seam/2.2.0.GA/reference/en-US/html/>

<http://docs.jboss.com/seam/latest/reference/en-US/html/gettingstarted.html>

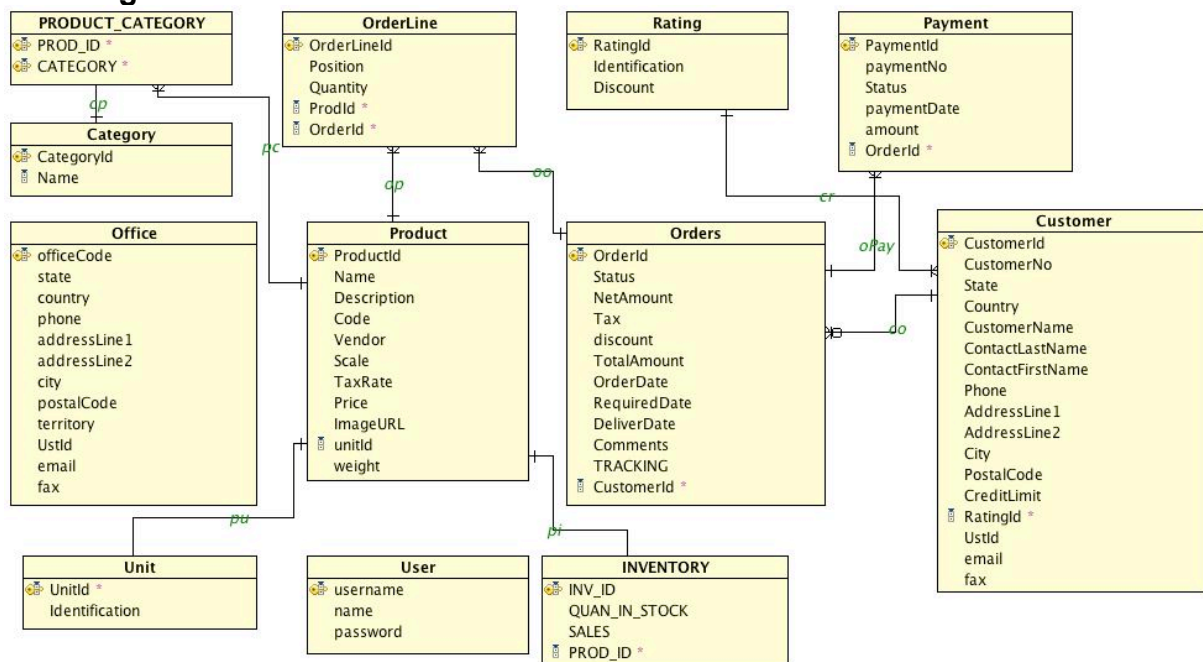
<http://docs.jboss.org/seam/1.1GA/reference/en/html/gettingstarted.html>

<http://javathreads.de/2008/09/tutorial-mit-jboss-seam-und-jee5-unter-eclipse-starten/>

Literatur:

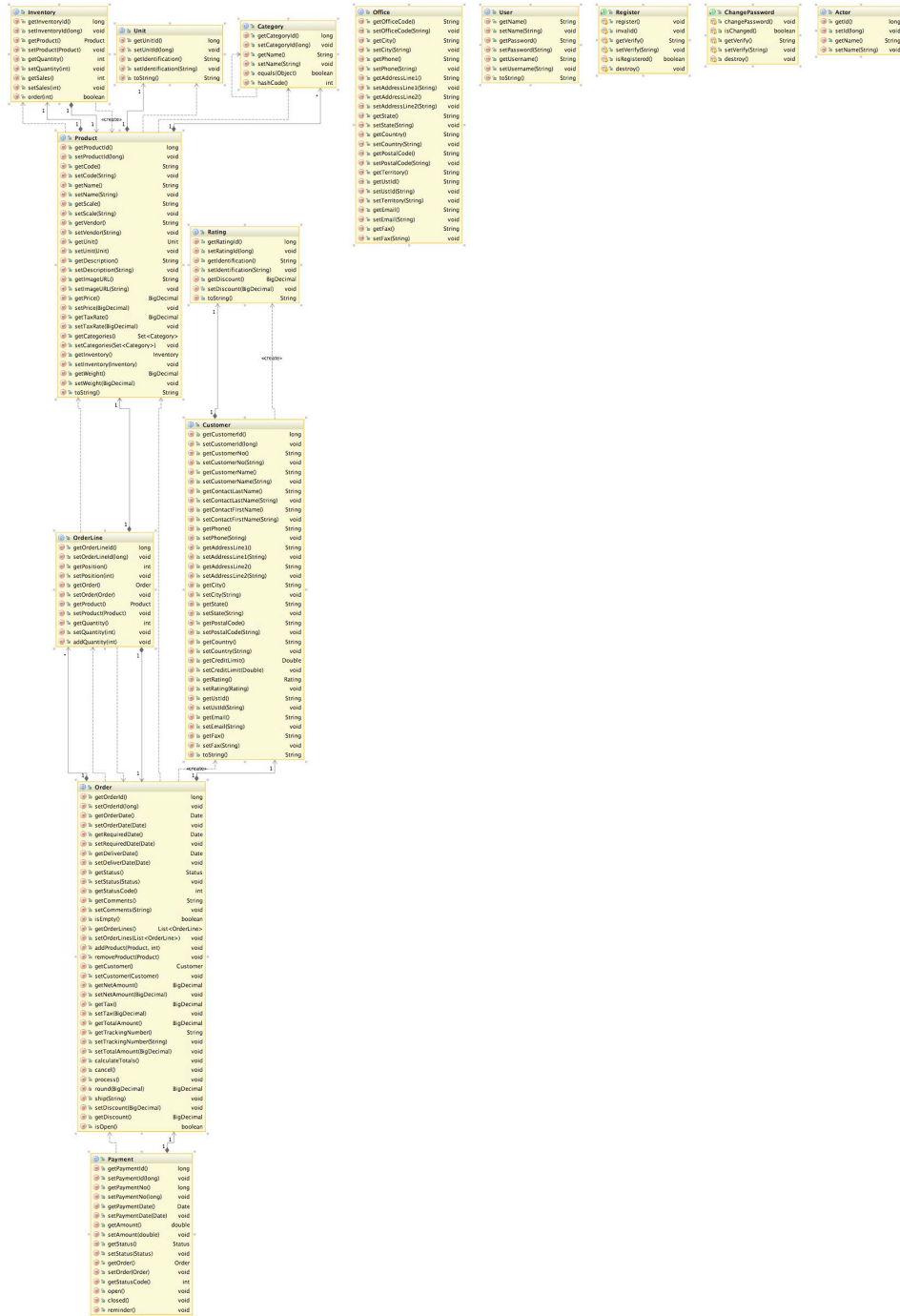
Allen Dan: Seam In Action, ISBN: 9781933988405, ISBN-10: 1933988401

## Beschreibung der Web-Applikation Arien ER-Diagram





# Entity-Diagram



## Locale Interface

ShoppingCart	
total	BigDecimal
cartSelection	Map
isEmpty	boolean
discount	BigDecimal
subtotal	BigDecimal
tax	BigDecimal
cart	List<OrderLine>
addProduct(Product, int)	void
updateCart()	void
resetCart()	void
destroy()	void

Shopping	
searchString	String
nextPageAvailable	boolean
selectedId	long
pageSize	int
searchPattern	String
selectEntity(Object)	void
find()	void
addToCart()	void
addAllToCart()	void
reset()	void
destroy()	void

ObjectBooking	
bookingValid	boolean
entityId	long
selectEntity(Object)	void
setBookingDetails()	void
confirm()	void
update()	void
delete()	void
insert()	void
cancel()	void
destroy()	void

Searching	
searchString	String
allValues	List<SelectedItem>
nextPageAvailable	boolean
allValues2	List<SelectedItem>
pageSize	int
searchPattern	String
find()	void
nextPage()	void
destroy()	void

Ship	
track	String
ship()	String
viewTask()	String
destroy()	void

StoreManager	
totalSales	BigDecimal
unitsSold	long
numberOrders	long
totalInventory	long

Accept	
accept()	String
reject()	String
viewTask()	String
destroy()	void

CurrentCustomer	
isEmpty	boolean
id	long
destroy()	void

Checkout	
createOrder()	void
submitOrder()	Order
destroy()	void

Authenticator	
authenticate()	boolean

Powered by yfiles

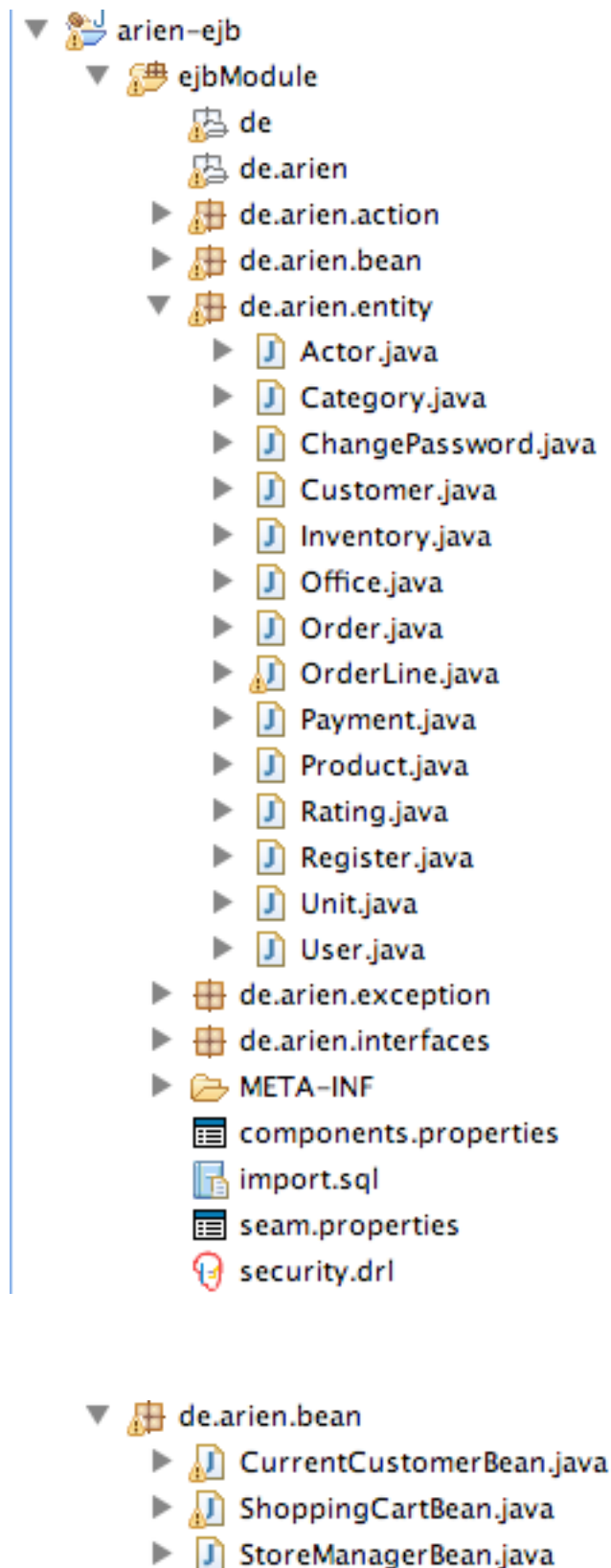
## Die Application

































Die Arien-Delikatessen-Applikation hat folgende Funktionalitäten.

- User registration
- Login
- Logout
- Set password
- Search in Tabela (Ajax)
- Shopping-Process
- Invoice-PDF
- ToDo-List-PDF

Das Programm benutzt JSF, EJB 3.0 and Seam, zusammen mit Facelets für View.  
Das Programm hat flg. Session-Beans für die Implementierung der Businesslogik.

### Die Entity-Beans implementieren der Application's persistente Domain-Model.



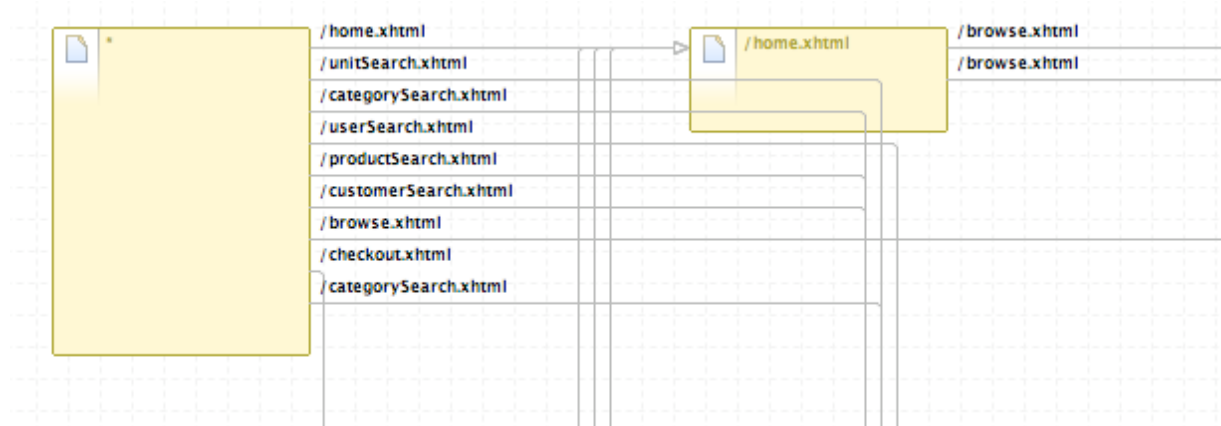
- ▼  de.arien.action
  - ▶  AcceptAction.java
  - ▶  AfterShippingAction.java
  - ▶  AuthenticatorAction.java
  - ▶  CategoryCrudAction.java
  - ▶  CategorySearchingAction.java
  - ▶  ChangePasswordAction.java
  - ▶  CheckoutAction.java
  - ▶  CustomerCrudAction.java
  - ▶  CustomerSearchingAction.java
  - ▶  InventoryCrudAction.java
  - ▶  InventorySearchingAction.java
  - ▶  OfficeCrudAction.java
  - ▶  OfficeSearchingAction.java
  - ▶  OrderApprovalDecision.java
  - ▶  OrderCrudAction.java
  - ▶  OrderLineCrudAction.java
  - ▶  OrderLineSearchingAction.java
  - ▶  OrderSearchingAction.java
  - ▶  PaymentCrudAction.java
  - ▶  PaymentSearchingAction.java
  - ▶  ProductCrudAction.java
  - ▶  ProductSearchingAction.java
  - ▶  RatingCrudAction.java
  - ▶  RatingSearchingAction.java
  - ▶  RegisterAction.java
  - ▶  ShipAction.java
  - ▶  ShoppingAction.java
  - ▶  UnitCrudAction.java
  - ▶  UnitSearchingAction.java
  - ▶  UserCrudAction.java
  - ▶  UserSearchingAction.java



## Die Program-Conversation

Die meisten Web-Applikationen verfügen in der Regel, nicht über ein Konstrukt zur Interaktion.

Entweder wird eine DB oder HttpSession eingesetzt, die abgesehen von einer mangelnden Skalierbarkeit sowie Latenz-Problemen, auch instabil bei den Funktionen „back“ und „New Window“ ist.



Seam bietet *conversation context* als erstes Klassen Konstrukt.

Schauen wir uns Customer-Search an:

Shopping | Reports | Customer | Payment | Order | Office | Unit | Rating | OrderLine | Product | Category | Inventory | User | Settings | Logout

Search Customer

Maximum results: 10

CustomerNo	Name	Phone	Fax	email	AddressLine1	City	PostalCode	CreditLimit	Rating	Discount	Action
103	XX	88987879	4435543	xx@xx.de	Wolfgang Str. 121	Ffm.	60322	2000.0	A	0.00	<a href="#">Customer</a>

Abbild CustomerSearch

```

@Stateful
@Name("customerSearch")
@Scope(ScopeType.SESSION)
@Restrict("#{identity.loggedIn}")
public class CustomerSearchingAction implements Searching
{
    @PersistenceContext
    private EntityManager em;

    private String searchString;
    private int pageSize = 10;
    private int page;
    private boolean nextPageAvailable;

    @DataModel
    private List<Customer> customers;

    @Logger
    private Log log;

    public void find()
    {
        page = 0;
        queryCustomer();
    }

    public void nextPage()
    {
        page++;
        queryCustomer();
    }

    private void queryCustomer() {
        log.info("queryCustomer : #{patternCustomer}");
        List<Customer> results = em.createQuery("select c from Customer c where lower(c.customerName) lik
        \"or lower(c.customerNo) like #{patternCustomer} \" )
        .setMaxResults(pageSize+1)
        .setFirstResult(page * pageSize)
        .getResultList();

        log.info("queryCustomer.results.size() : " + results.size() );
    }
}

```

Der @Stateful Annotation identifiziert dieses class als ein stateful session bean.

Der @Restrict Annotation ist eine security Restriction des Komponentes.

Der @Factory(“”) Annotation erlaubt ein Seam component als Initiator für ein non-component object zu agieren.

Der @DataModel Annotation exponiert eine List als JSF ListDataModel. In CustomerSearch die liste der Customers ist in die variable named customers ausinjeziert (Outjection).

Der @Remove Annotation besagt, dass der spezifizierte stateful session bean soll entfernt und sein state nach der Aufruf destroyed werden.

## In CustomerSearch.xhtml

```
<h1>Search Customer</h1>

<h:form id="searchCriteria">
<fieldset>
  <h:inputText id="searchString" value="#{customerSearch.searchString}" style="width: 165px;">
    <a:support id="onkeyup" event="onkeyup" actionListener="#{customerSearch.find}" reRender="searchResults" />
  </h:inputText>
  &#160;
  <a:commandButton id="findCustomer" value="Find Customer" action="#{customerSearch.find}" reRender="searchResults"/>
  &#160;
  <a:commandButton id="insertCustomer" value="Insert Customer" action="/insertCustomer.xhtml" reRender="insertCustomer"/>
  &#160;
  <a:status id="status">
    <f:facet id="StartStatus" name="start">
      <h:graphicImage id="SpinnerGif" value="/img/spinner.gif"/>
    </f:facet>
  </a:status>
  <br/>
  <h:outputLabel id="MaximumResultsLabel" for="pageSize">Maximum results:</h:outputLabel>&#160;
  <h:selectOneMenu id="pageSize" value="#{customerSearch.pageSize}">
    <f:selectItem id="PageSize5" itemLabel="5" itemValue="5"/>
    <f:selectItem id="PageSize10" itemLabel="10" itemValue="10"/>
    <f:selectItem id="PageSize20" itemLabel="20" itemValue="20"/>
  </h:selectOneMenu>
</fieldset>
```

Der RichFaces Ajax `<a:support>` tag erlaubt ein JSF action event listener die Ausführung eines asynchronous XMLHttpRequest, wenn ein JavaScript event wie ZBs. onkeyup ausgelöst wird. Der `reRender` attribute erlaubt die partielle rendering und update, wenn der asynchronous response auftritt.

Der RichFaces Ajax `<a:status>` tag illustriert die Wartezustand für asynchronous requests.

Der RichFaces Ajax `<a:outputPanel>` tag definiert die Region der Seite, die bei an asynchronous request re-rendere werden kann.

Jetzt schauen wir uns die Umstezung von conversation-scoped stateful session an, am Beispiel einer CustomerCRUDAction :

## Customer

<b>CustomerNo:*</b>	<input type="text" value="103"/>
<b>State:*</b>	<input type="text" value="Hessen"/>
<b>Country:*</b>	<input type="text" value="Germany"/>
<b>Name:*</b>	<input type="text" value="XX"/>
<b>LastName:*</b>	<input type="text" value="YYY"/>
<b>FirstName:*</b>	<input type="text" value="x"/>
<b>Phone:</b>	<input type="text" value="88987879"/>
<b>Fax:</b>	<input type="text" value="4435543"/>
<b>E-Mail:</b>	<input type="text" value="xx@xx.de"/>
<b>AddressLine1:*</b>	<input type="text" value="Wolfgang Str. 121"/>
<b>AddressLine2:</b>	<input type="text"/>
<b>City:*</b>	<input type="text" value="Ffm."/>
<b>PostalCode:*</b>	<input type="text" value="60322"/>
<b>CreditLimit:*</b>	<input type="text" value="2000.0"/>
<b>RatingId:</b>	<input type="text" value="A"/>
<b>UstId:*</b>	<input type="text" value="193936932"/>

[Update](#)[Delete](#)[Cancel](#)

```

@Stateful
@Name("customerCRUD")
@Restrict("#{identity.loggedIn}")
public class CustomerCrudAction implements ObjectBooking {
    private static final long serialVersionUID = 589289832

    @PersistenceContext(type=PersistenceContextType.EXTEND
    EntityManager em;

    @Resource
    SessionContext ctx;

    @In
    Context sessionContext;

    @In(create=true)
    @Out
    Customer customer;

    @In
    private FacesMessages facesMessages;

    @In
    private Events events;

    @Logger
    private Log log;

    private boolean bookingValid;
    private long ratingId;

    @Begin(join=true)
    public void selectEntity(Object selected) {
        customer = em.merge((Customer) selected);
    }

    public void setBookingDetails() {
        bookingValid = true;
    }

    public boolean isBookingValid() {
        return true;
    }

    @End
    public void update() {

```

CustomerCRUDAction benutzt ein EJB3 *extended persistence context*, so dass, jede Entity- Instance für die gesamte lifecycle der stateful session Verantwortlich ist.

Der [@Out](#) Annotation deklariert, dass der Wert der Attributes ist *outjected* wrden muss,

Der [@Begin](#) Annotation ein *long-running conversation* spezifiziert, dh. die aktuelle conversation context wird nich nach der Request-Ende destroyed werden soll. Im Gegenteil der soll reassociatiert werden mit jegliche request von der aktuelle window, bis die Annotation `@End` Methode erreicht ist.

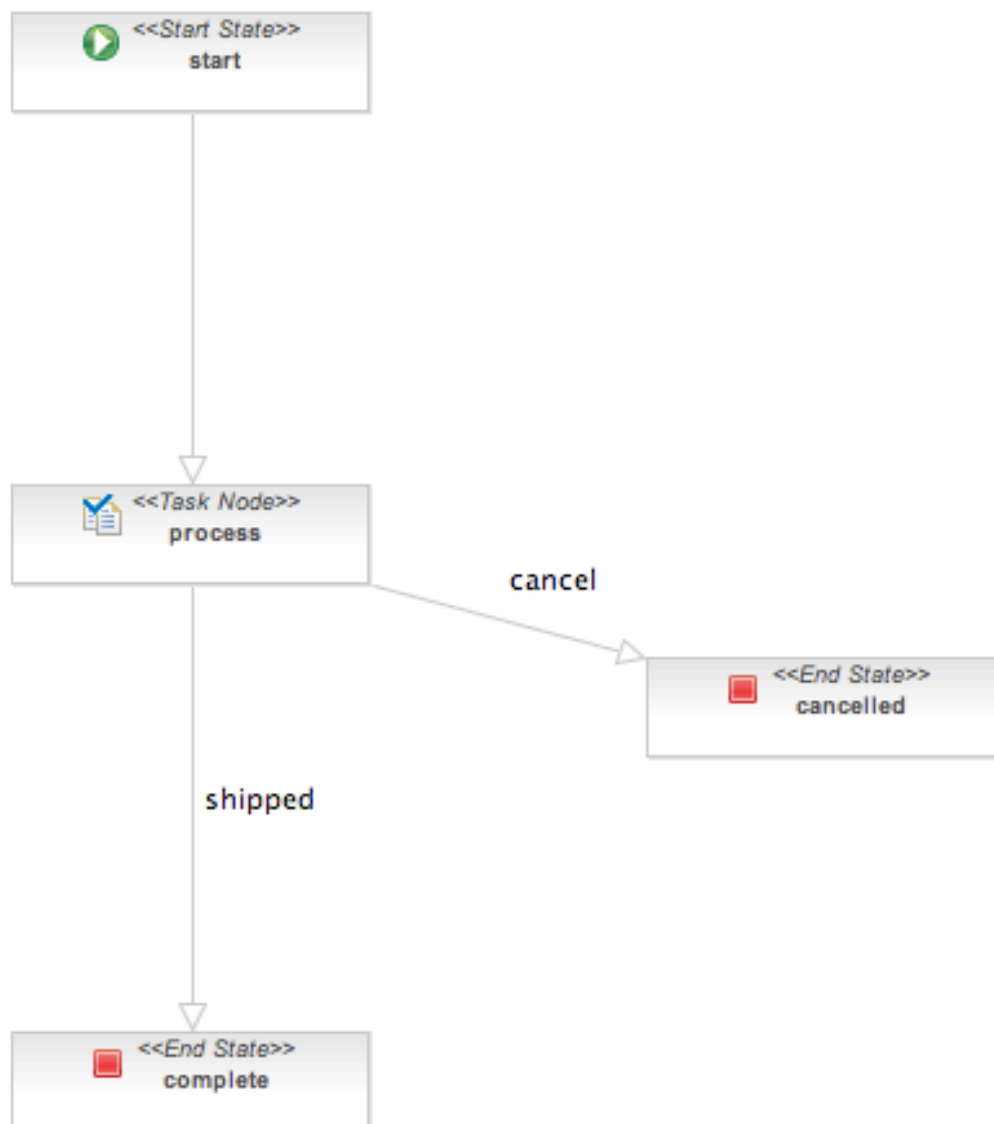
Der [@Begin\(nested=true\)](#) verursacht, dass ein nested conversation in conversation stack gepushed wird dh. alle Komponen haben Zugriffsmöglichkeiten auf outer conversation state, aber jeglich veränderung der Innerzustand hat keine Effekt auf Äussere.

Der `@remove` Methode wird nach der destroy der Conversation ausgeführt.

## ***Business process management in Seam***

Seam's jBPM Integration mach die Task-Managemt einfach.

**Prozessdefintion in Checkout.jpdl.xml**



```

<pageflow-definition
  xmlns="http://jboss.com/products/seam/pageflow"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation=
    "http://jboss.com/products/seam/pageflow http://jboss.com/products/seam/pageflow-2.2.xsd"
  name="checkout">
  <start-state name="start">
    <transition to="confirm"/>
  </start-state>

  <page name="confirm" view-id="/confirm.xhtml"
    no-conversation-view-id="/checkout.xhtml">
    <redirect/>
    <transition name="cancel" to="cancel"/>
    <transition name="purchase" to="complete">
      <action expression="#{checkout.submitOrder}" />
    </transition>
  </page>

  <page name="complete" view-id="/complete.xhtml">
    <end-conversation/>
    <redirect/>
  </page>

  <page name="cancel" view-id="/browse.xhtml">
    <end-conversation before-redirect="true"/>
    <redirect/>
  </page>
</pageflow-definition>

```

## Reports

Seam ermöglicht Hybernate basierte BIRT-Frame zur Erstellung der PDF/XSL/CSV-Reports, dh einmalige implementation der Businesslogik.

Firma  
YY  
Wolfsgang Str. 121  
D-60322-Ffm.

Kunden Nr.: 105  
Steuernr. : 193936932  
Lieferdatum: 30.09.2009  
Datum: 20.10.2009

Rechnung Nr. 1051

Position	Menge	Art.Nr	Beschreibung	Einzelpreis €	Gesamtpreis €
1	1 Verpackung: verschießbare Frischbehälter	2104	rote Paprika gefüllt mit Frischkäse, Doppelrahmkäse, Knoblauch, Gewürze, Kräuter, Pflanzenöl und Salz.	12,00	9,60
Gesamt Netto					9,60
zzgl. 8 % USt. auf					0,77
Discount 0 %					4,80



## Die direkte Einbindung des Reportes in xhtml-File

```
<p:birt xmlns:ui="http://java.sun.com/jsf/facelets"
        xmlns:s="http://jboss.com/products/seam/taglib"
        xmlns:p="http://jboss.com/products/seam/birt"
        designType="run"
        format="pdf"
        locale="de_DE"
        designName="invoice.rptdesign"
        title="Arien Report">
</p:birt>
```

es besagt, dass beim Aufruf soll die invoice.rptdesign mit der Locale de\_DE ausgeführt und das Ergebnis soll als PDF-File vorliegen.